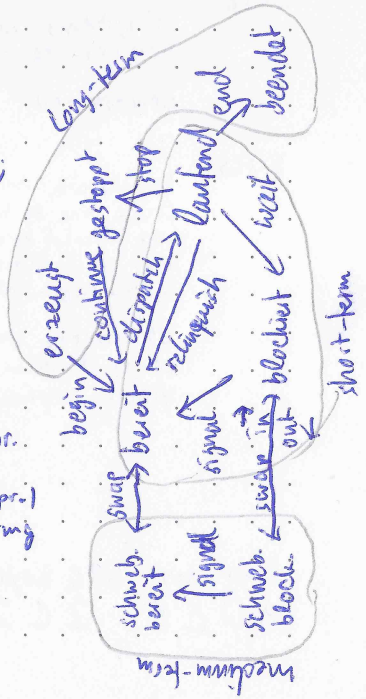
```
qsort(arr, n, sizeof(*arr), cmp);
while (fgets(buf, sizeof(buf), stdin)) {}
if (ferror(stdin)) die();
counter = 0;
par[counter++] = strtok(buf, " \n");
while ((par[counter++] = strtok(NULL, " \n")));
execv[lv]p? // list, vector, PATH
struct stat buf;
if (stat(file, &buf) < 0) die();
S_IS{REG, DIR, LNK, FIFO?}(buf.st_mode)
DIR *dp = opendir(path);
if (!dp) die();
struct dirent *e;
while (errno = 0, e = readdir(dp)) {
    e->d_name
    errno = 0; break;
}
if (errno) die();
if (closedir(dp)) die();
void *thread_func(void *arg) {
    errno = pthread_detach(pthread_self());
    return res;
}
pthread_t threads[n];
if (!errno = pthread_create(&threads[i], NULL,
    thread_func, (void*) arg)) die();
if (!errno = pthread_join(threads[i], &res)) die();
int sockfd, err;
struct addrinfo *ailist, *aip;
struct addrinfo hint = {
    .ai_socktype = SOCK_STREAM,
};
if ((err = getaddrinfo("host", "42", &hint, &ailist))) {
    if (err == EAI_SYSTEM) die();
    else fprintf(stderr, "%s\n", gai_strerror(err));
}
for (aip = ailist; aip; aip = aip->ai_next) {
    if ((sockfd = socket(aip->ai_family, aip->
        ai_socktype, aip->ai_protocol)) < 0) {
        perror("socket"); continue;
    }
    if (!connect(sockfd, aip->ai_addr, aip->ai_
        addrlen)) break;
    perror("connect");
    if (close(sockfd)) perror("close");
}
freeaddrinfo(ailist);
if (!aip) fprintf(stderr, "failed to conn");
int sockfd = socket(AF_INET6, SOCK_STREAM, 0);
if (sockfd < 0) die();
struct sockaddr_in6 a = {
    .sin6_family = AF_INET6,
    .sin6_port = htons(port),
    .sin6_addr = in6addr_any,
};
if (bind(sockfd, &a, sizeof(a)) < 0) die();
if (listen(sockfd, SOMAXCONN) < 0) die();
for (;;) {
    int client = accept(sockfd, NULL, NULL);
    if (client < 0) die();
}
struct sigaction action = {  // zombies
    .sa_handler = SIG_IGN,
};
sigemptyset(&action.sa_mask);
sigaction(SIGCHLD, &action, NULL);
static void sahandler(int sig) {
    // errno sichern
}
struct sigaction sa = {
    .sa_handler = sahandler,
    .sa_flags = SA_RESTART | SA_NOCLDSTOP,
};
sigemptyset(&sa.sa_mask);
sigaction(SIGCHLD, &sa, NULL);
sigset_t oldm, newm;
sigemptyset(&newm);
sigaddset(&newm, SIGCHLD);
static volatile bool cond;
while (!cond) {}
sigprocmask(SIG_BLOCK, &newm, &oldm);
while (!cond)
    sigsuspend(&oldm);
sigprocmask(SIG_SETMASK, &oldm, NULL);
pthread_cond_t cond;
pthread_mutex_t mutex;
errno = pthread_cond_init(&cond);
errno = pthread_mutex_init(&mutex);
pthread_mutex_lock(&mutex);
pthread_cond_wait(&cond, &mutex);
pthread_cond_broadcast(&cond);
pthread_mutex_unlock(&mutex);
pthread_cond_destroy(&cond);
pthread_mutex_destroy(&mutex);
do {} while (!__sync_bool_compare_
    and_swap(&mem, old, new));
freopen("file", "mode", stdin);
if (dup2(fileno(f), STDIN_FILENO) < 0) die();
close(fileno(f));
int status;
pid_t child;
while ((child = waitpid(-1, &status, WNOHANG)) > 0) {
    WIFEXITED(status)
    WEXITSTATUS(status)
}
if (child < 0) die();
if (kill(pid, SIGTERM) < 0) die();
```

Stack
↓
↑
Heap
BSS init = 0
Data init ≠ 0
Text

5 problemorientierte Sprachebene
cc
4 Assemblersprachebene
as, ld
3 Maschinenprogrammebene
partielle Interpr. (OS)
2 Befehlssatzebene
Interpr. [Mikroprogr.] oder Ausführung
1 Mikroarchitekturebene
Ausführung
0 digitale Logikebene



STRING(3D) STRING(3D)

NAME
strfry - string operation

SYNOPSIS
#include <string.h>
char *strfry(s1,s2)
char *s1,*s2;

DESCRIPTION
The arguments s1 and s2 point to strings (arrays of characters terminated by a null character). The function strfry may or may not alter s2 or s1. This function does not check for overflow of the array pointed to by s1.

strfry will encrypt s1 using s3 as the key. (s3 is a character pointer and contains random garbage from the stack.) s2 will then be copied to the memory pointed to by the Null pointer. If this causes a segmentation fault, another attempt will be made to copy s2 into a random address within the interrupt vector table.

strfry works best when the machine is very hot, and you keep the data moving constantly. Unless your memory devices are teflon coated

NOTE
In systems where strfry is installed, make certain permissions are set as shown for /dev/kmem:

crw-rw-rw- 1 root sys 0, 0 May 6 /dev/kmem

BUGS
In certain machine architectures strfry doesn't always crash the system in the first attempt. In these systems, you should execute it in a loop at least three times. If this still fails use the inline assembler to insert a halt-and-catch-fire (HLF) instruction into the code.

Character movement is performed differently in different implementations. Thus overlapping moves may yield surprises.

STRING(3D)

Sign my GPG key: 4096R/9021D06D 2014-01-20 Julian Brost <julian@0x4a42.net>
ADC5 2C2B FEA6 DA44 5904 5D70 22DC 613E 9021 D06D